CMPT 117

Computer Science Final Exam

April 14, 2003, 9am

This exam is open book. You may bring in up to 100 pounds of paper-based support materials, but no surrogates or computers. You may bring in a calculator.

Please read the question carefully before answering. *You cannot be given marks for answering the wrong question.* A portion of the marks will be awarded for the style and clarity of your answer. Answer all questions in the separate exam booklet provided.

1. Short Answer (25 marks)
   a. (2 marks) What is the STL?
   b. (2 marks)What is an *iterator?*
   c. (2) What is a *map*?
   d. (2) Give the short cut key, or draw the icon, for setting a breakpoint and running a Visual C++ program in Debug Mode.
   e. (2) What is the *best* case timing for selection sort?
   f. (2) What is the *worst case* timing for quicksort?
   g. (4) Under what circumstances is the worst case for quicksort equal to the best case for bubblesort?
   h. (2) Simplify O($3n^2 + 14n + 87$).
   i. (4) Comlete the following: A pointer is a(n) _______________ representing a _______________ _______________. It takes up _______________ bytes of memory to store.
   j. (3)If I'm writing the iterative traversal/print for a doubly linked list, what is wrong with the following code?

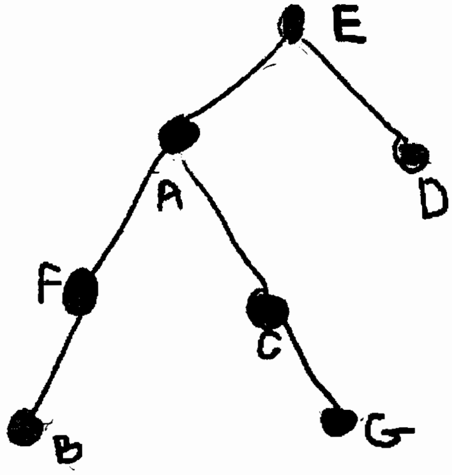
```
dlink *temp;
temp = Head;

while(temp!=0) {
   cout << *temp;
   temp++;
}
```

?? 2. (15 marks) What is the output? Assume objects are destroyed in the order they are declared.

```
#include <iostream>
using namespace std;

class Myclass {
public:
      Myclass()
      {
            cout << "default constructor" << endl;
            _data = 3;
      }
      Myclass(int i)
      {
            cout << "int constructor input " << i << endl;
            _data = i;
      }
      Myclass(const Myclass &MC)
      {
            cout << "copy constructor input " << MC._data<< endl;
            _data = MC._data + 2;
      }
      ~Myclass() { cout << "destroying " << _data << endl;}
      Myclass& operator=(const Myclass& M)
      {
            cout << "assignment of " << M._data << endl;
            _data = M._data - 1; return *this;
      }
      int _data;
};

Myclass myfunc0( const Myclass &In0 )                  _data=3
{
      Myclass temp; — default constructor
      temp = In0; — operator =
      return temp;
}

Myclass & myfunc1( const Myclass In1 )
{
      Myclass *temp = new Myclass(In1);
      return *temp;
}

int main()
{
      cout << "one..." << endl;
      Myclass M(3);

      cout << "two..." << endl;
      Myclass M2 = myfunc0( M );

      cout << "three..." << endl;
      Myclass M3( myfunc1( M2 ) );
      return 0;
}
```

One...
int constructor input 3
two...
default constructor
assignment of 3
default constructor
assignment of 2
three...
Copy constructor input !
int constructor input 3
destroying 3

3. Tree traversals (10 marks)



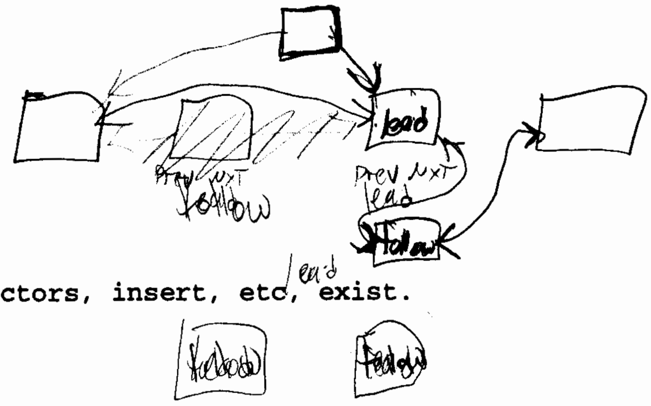
The diagram shows an *unsorted* binary tree rooted at node E. Recall the definitions of inorder, preorder and postorder traversals:

a. An inorder traversal visits the left subtree, prints the value of the current node, then visits the right subtree
b. A preorder traversal prints the value of the current node, visits the left subtree, then visits the right subtree.
c. A postorder traversal visits the left subtree, then visits the right subtree, the prints the contents of the node.

Give the inorder, preorder and postorder traversals of the above tree.

4. (15 marks) Below is a definition of a dLinkedList class. It contains a *struct* node definition. Assume that the *prev* pointer of the *Head* node, and the *next* pointer of the last node are null.
Implement the sort member, sorting the elements in ascending order. You can use any sorting algorithm you wish.

```
class dlinkedList {
public:
   struct dlink {
      dlink *prev;
      int data;
      dlink *next;
   };
   // assume all constructors, insert, etc, exist.
   void sort();
   dlink *_Head;
};
```



5. (15 marks)
Design a vector class exactly like the vector class in the STL, except that it also has a method called *doubleup.*

Your solution should include a templated definition of *doubleup* that takes each element of the vector and multiplies it by 2.

For the case where the vector contains characters, *doubleup* should convert all characters to uppercase. You can use the builtin function *toupper(char)* to do this for you.

For the case where the vector contains strings, *doubleup* should double up the string. So, if the vector initially contained strings with values {"cat", "dog", "horse"}, it should contain {"catcat", "dogdog", "horsehorse"} afterwards.

6. **(10 marks)**
What is the output of the following program?

```
#include <iostream>
using namespace std;

int main()
{
   int ia[ 10 ] = { 0, 2, 4, 6, 8, 10, 12, 14, 16, 18 };
   int j;
   int *ip = ia;

   for ( j = 1; j < 8; j += 2 )
      cout << *( ip + j ) << endl;
  for ( ;  j > 0; j -= 2 )
      cout << ip[ j ] << endl;

   return 0;
}
```

ia

ip = ia

7

ip[[illegible]]

7. (10 marks) Write a recursive program that counts the number of elements in a tree.

count
0
1
2
3